In the United States Patent and Trademark Office

Applicant:	David R. Baldwin
Application No.:	09/591,225
Filing Date:	June 9, 2000
Examiner:	Richer, Aaron M.
Title:	Graphic Memory Management with Invisible Hardware-Managed Page Faulting
Art Unit:	2671
Docket No.:	TD-155

For: Graphics Memory Management With Invisible Hardware-Managed Page Faulting

APPEAL BRIEF

Honorable Commissioner of Patents and Trademarks
Alexandria, VA 22313

Sir:

Enclosed is an Appeal Brief with three Appendices. Docketing for Oral Argument is requested.

Any extension of time necessary for consideration of this appeal is also hereby requested. However, the Commissioner is authorized to charge any fees, or credit any overpayment, to Deposit Account Number 07-2320.

Table of Contents

Table of Authorities

Statutes
35 USC 103(a)

Foreign Patents
Kaiser (European Patent Publication 0766177)

Non-patent Literature
Jim Blinn's Corner, "The Truth About Texture Mapping"
"Nachos" document photocopied by Examiner

Real Party in Interest

The real party in interest, and assignee of this case, is *3Dlabs Inc., Ltd.*, of Reid Hall, Hamilton HM11, Bermuda, which is owned by *Creative Labs, Inc.* headquartered in Singapore.

Related Appeals and Interferences

To the best knowledge and belief of the undersigned attorney, there are no related appeals or interferences.

Status of Claims

Claims 1-5, 7-10, and 12-22 are currently pending in the present application, and stand rejected. These claims are appealed. Claims 6 and 11 were cancelled in a previously filed amendment.

Status of Amendments

No amendments were submitted in response to the Examiner's final Office actin.

Summary of Claimed Subject Matter

The following summary refers to disclosed embodiments and their advantages but does not delimit any of the claimed inventions.

The claimed subject matter of independent claim 1 includes a computer system (such as shown in FIG. 1, in totality), a graphics accelerator (e.g., page 8, line 12; FIG. 3, etc.), which managed page faulting of texture data invisibly to the host processor (page 8, lines 11-15).

Independent claim 2 recites a computer system (FIG. 1), a graphics accelerator unit (page 8, line 12; FIG. 3) which manages page faulting of texture data from main memory (texturing is discussed at page 2, line 20) from main memory (FIG. 1, 460) used by at least one host processor (FIG. 1, 425) into a dedicated graphics memory (FIG. 3, SGRAM/SDRAM), invisibly to the host processor (page 8, lines 11-15), except when said graphics accelerator unit calls for data which has not recently been presenting said main memory (page 8, lines 17-25).

Independent claim 3 claims a computer system (FIG. 1) with at least one CPU (FIG. 1, 425) having access to a main memory (RAM 460), first memory logic unit (427) which virtualizes the main memory with reference to at least one bulk storage device (435), a graphics accelerator unit (page 8, line 12; FIG. 3) comprising rendering logic, dedicated graphics memory, and a second memory management unit (discussed at page 3, and page 4, line 5) which manages texture data for the accelerator logic and performs page faulting invisibly to the CPU.

Independent claim 4 includes a computer system (FIG. 1) comprising a host processor (FIG. 1, 425) having physical memory (FIG. 1, 460), a graphics accelerator unit (FIG. 3) having rendering logic (page 3, line 27) dedicated graphics memory (FIG. 3, SGRAM/SDRAM) and a memory management unit (Discussed at page 3, and page 4, line 5), wherein when said graphics accelerator unit attempts to access texture data from the physical memory associated with the host, the graphics memory manager fetches the texture data automatically (page 8, lines 10-16).

Independent claim 7 includes a computer system (FIG. 1) with a host processor (FIG. 1, 425) and host memory (RAM 460) and having virtual memory management (page 3, starting at line 30) and a graphics accelerator unit (FIG. 3) having physical memory (SGRAM/SDRAM, FIG 3) and having virtual memory management (page 3, starting line 30) wherein when the graphics accelerator attempts to access texture data from the host physical memory, the accelerator unit fetches the data automatically (page 8, lines 10-16); and if said texture data isn't in

the physical memory, loading into the host physical memory and thereafter automatically fetching it (page 8, lines 17-29).

Independent claim 14 includes a computer system (FIG. 1) with a CPU (FIG. 1, 425) connected to a main memory (RAM 460) with first memory management logic which virtualizes main memory (page 3, starting at line 30). A bulk storage unit (ROM 453) and graphics accelerator unit (FIG. 3) with accelerator logic (FIG. 3) dedicated graphics memory (SGRAM/SDRAM, FIG. 3) and second memory management logic (see memory management at page 3, starting at line 30) which manages texture data for the accelerator logic and performs page faulting of the texture data invisibly to the host processor (page 8, lines 10-16), wherein the graphics accelerator includes PCI/AGP, DMA, SGRAM/SDRAM, a RAMDAC and a video stream interface (see variously FIG. 3).

Independent claim 15 includes a computer system (FIG. 1) with a CPU (FIG. 1, 425) connected to a main memory (RAM 460) with first memory management logic which virtualizes main memory (page 3, starting at line 30). A bulk storage unit (ROM 453) and graphics accelerator unit (FIG. 3) with accelerator logic (FIG. 3) dedicated graphics memory (SGRAM/SDRAM, FIG. 3) and second memory management logic (see memory management at page 3, starting at line 30) which manages texture data for the accelerator logic and performs page faulting of the texture data invisibly to the host processor (page 8, lines 10-16).

Independent claim 17 includes a computer system (FIG. 1) comprising a host processor (FIG. 1, 425) having physical memory (FIG. 1, 460), a graphics accelerator unit (FIG. 3) having rendering logic (page 3, line 27) dedicated graphics memory (FIG. 3, SGRAM/SDRAM) and a memory management unit (Discussed at page 3, and page 4, line 5), wherein when said graphics accelerator unit attempts to access texture data from the physical memory associated with the host, the graphics memory manager fetches the texture data automatically (page 8, lines 10-16), wherein the graphics accelerator includes PCI/AGP, DMA, SGRAM/SDRAM, a RAMDAC and a video stream interface (see variously FIG. 3).

Independent claim 20 includes a computer system (FIG. 1) with a host processor (FIG. 1, 425) and host memory (RAM 460) and having virtual memory management (page 3, starting at line 30) and a graphics accelerator unit (FIG. 3) having physical memory (SGRAM/SDRAM, FIG 3) and having virtual memory management (page 3, starting line 30) wherein when the graphics accelerator attempts to access texture data from the host physical memory, the accelerator unit fetches the data automatically (page 8, lines 10-16); and if said texture data isn't in the physical memory, loading into the host physical memory and thereafter

automatically fetching it (page 8, lines 17-29), wherein the graphics accelerator includes PCI/AGP, DMA, SGRAM/SDRAM, a RAMDAC and a video stream interface (see variously FIG. 3).

Independent claim 22 includes a graphics accelerator unit (FIG. 3) and a memory management unit (FIG. 3, memory management unit) which maintains texture data for the accelerator logic and performs page faulting of texture data (see page 8, lines 10-16) wherein said accelerator determines where a page is located in host physical memory, determines which page from working set to use, marks this page as most recently used, and updates the page tables and removes any reference to the bumped page (see page 8, lines 17-30).

Grounds of Rejection to be Reviewed on Appeal

Whether Claims 1-5, 7-10, and 12-22 rejected under 35 USC 103(a) as unpatentable over Kaiser (European patent publication 0 766 177) in view of Blinn (Jim Blinn's Corner "The Truth About Texture Mapping").

Arguments

I. First Ground of Rejection: Claims 1-5, 7-10, and 12-22 rejected under 35 USC 103(a) as unpatentable over Kaiser (European patent publication 0 766 177) in view of Blinn (Jim Blinn's Corner "The Truth About Texture Mapping").

All claims are rejected under this single ground for rejection. However, several of the claims are separately patentable and are argued separately, under the appropriate headings below.

A. Claims 1, 2, 4, 5, 7-10, 12, 13, and 16-22.

The disagreement in this case involves, at least in part, the definition of a "page fault" as used in the claims of the present application, and in the references cited by the Examiner against the present application.

Applicant's position on this points out that a "page fault" is defined in the present application and in the Kaiser reference, which Examiner primarily relies on (in combination with other references, as described more fully below). The present application and the Kaiser reference both define a page fault in the same way.

The term "page fault" is used, for example, in claim 1 of the present application:

> 1. A computer system, comprising:
>
> a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor.

1. Virtual address translation is not the same as a page fault. Kaiser only teaches virtual address translation without host processor involvement. Kaiser explicitly requires the host processor to handle a page fault. Hence, Kaiser does not teach or suggest managing page faulting invisibly to the host processor, as claimed in at least claim 1.

In rejecting claim 1, Examiner states:

> As to claim 1, Kaiser discloses a graphics accelerator unit which manages page faulting of graphics data invisibly to the host processor. Col. 3, lines 50-col. 4, line 23 and col. 5, lines 37-58 describe the

> address translation units, in a memory controller, with graphics engine, which manage page faults separate from, and invisible to, the host processor as long as the page being accessed is in a second level, or physical memory. This is similar to how the page faulting system claimed by applicant is described on pages 8-9 of applicant's specification. Col. 2, lines 1-27 explain the reasoning for making such page faults invisible to the CPU.

Page faulting is described in the present application, such as on page 8, in the summary of the invention. That passage states in part:

> When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory) it will be fetched in automatically by the graphics memory manager, and the host is not aware anything has happened.

The description and example of a page fault given in the present application necessarily means that the unit seeking the data has discovered that the data is not currently in that higher level of memory. Hence, the data is sought in a lower level of memory, such as the host's physical memory.

In situations where a unit other than the host processor attempts to access data with a virtual address (such as in Kaiser, where a memory controller attempts to access virtual data), when a page fault occurs, the host processor is informed, and the host processor resolves the page fault by fetching the required data from a lower level of memory. This costs the host processor overhead as it performs the tasks associated with managing the page fault. In the present application, Applicants claim (in claim 1) a computer system where a graphics accelerator (and not the host processor) manages page faulting of texture data, without requiring the host processor to manage the page fault. This saves the host processor's capacity for other functions.

This definition of a page fault is common and well established. In support of this, Applicant points to the primary reference cited by Examiner, namely, the Kaiser reference.

Kaiser describes a system wherein the memory controller includes a command buffer for storing a command block, a translation lookaside buffer (TLB), and a page table buffer. See, for example, Kaiser's abstract. The innovations of Kaiser are directed toward translating virtual addresses contained in command blocks (such as commands coming from the host processor). Kaiser at col. 3, lines 50-56. Kaiser automatically performs virtual address translation (not page faults), without host processor involvement. Kaiser at col. 4, lines 5-14. See also Kaiser at col. 4, lines 15-20, which state:

> It is an advantage of the present invention that 3D graphics processing may be efficiently accomplished by an auxiliary function processor in a controller connected to the processor bus, without the overhead of the processor translating the addresses in the command blocks to real addresses....

[Emphasis added.] Hence, Kaiser only removes host processor involvement in address translation. Kaiser does not remove host processor involvement from page faults. Kaiser explicitly states that when a page fault occurs, the host processor must be involved.

After describing how Kaiser deals with address translation, it then describes how a page fault must be handled by the host processor (which Kaiser refers to as processor 12). Col. 6 line 58-col. 6 line 14 state:

> If no entry is found in the system page table 226, a page fault condition is sent to the graphics engine 206. Since memory controller 204 is not a processor, it cannot deal directly with a page fault. If a page fault condition occurs while graphics processor 206 is processing a command block from the main processor complex 12, an error status packet reflecting the page fault condition is passed to the processor 12 by memory controller 204. Processor software recognizes the fault condition and causes the faulting address to be rerun. When processor 12 encounters the same page fault condition, system software (outside the scope of the present invention) resolves the condition and passes control back to the graphics processor 206.

[Emphasis added.] Hence, Kaiser explicitly states that the host processor must be involved in the event of a page fault. This distinguishes it from the language of claim 1 of the present application.

Both the application's definition of page fault and Kaiser's definition of page fault support Applicant's position. Kaiser explicitly says it requires host processor involvement when a page fault occurs. Various claims of the present application claim no host processor involvement in a page fault (e.g., texture data is fetched "automatically" or "invisibly" to the host processor).

Applicant respectfully submits that Examiner has conflated Kaiser's virtual address translation (which occurs using a TLB used by a memory controller) with page faults (which occur when the page table of the host processor does not have an address for the data). As shown above, Kaiser requires host processor involvement when a page fault occurs. Kaiser, however, does not require host processor

involvement when a virtual address translation is needed. In both the present application and in Kaiser, address translation and page faults are distinguished and are referred to as different actions, which in fact they are. However, Examiner's position is that Kaiser's teaching of host processor independent virtual address translation is the same as Applicant's claim 1, which deals not with virtual address translation, but with page faults.

Accessing virtual data in Kaiser requires two separate steps: First, the virtual address must be translated into a real address, so the system or application asking for the data knows where that data should be stored. This step is normally performed by the host processor, but Kaiser teaches that it can be performed without the host processor. Second, the system or application accesses those addresses and retrieves the data.

A page fault may occur only on the second step, and is defined as when the data is not in fact stored where the virtual address translation says it should be (for example, because the virtual address has been overwritten or is stale for some reason). At the time of virtual address translation (the first step), it is not known whether a page fault will occur or not (the second step). However, Examiner is interpreting the first step as the page fault. Examiner's interpretation directly contradicts Kaiser, which states:

> The address translation logic includes a four-way translation lookaside buffer (TLB) 220. If there is a miss indicated by a no compare from compare circuit 222 between addresses in TLB 220 and the effective address presented by graphics processor 206, memory controller performs a table walk of page tables in memory system 24.... Memory controller 204 fetches cache lines from the system page table on an as-needed basis to find an effective to real address translation. If no entry is found in the system page table 226, a page fault condition is sent to the graphics engine 206.

[col. 5 line 45-col. 6 line 2]

The above passage, summarized, says that virtual addresses are translated using a TLB. If the address translation is not in the TLB, then the memory controller of Kaiser goes to the host's page tables and finds an updated address translation. Kaiser's memory controller finds this updated address translation without host processor involvement--this is Kaiser's innovation. However, if the address translation is not in the system page tables, then the data is not in that level of memory and must be fetched from a lower level of memory. Both Kaiser and the present application refer to this condition as a page fault. Kaiser requires host

processor involvement during a page fault, as shown above. Hence, Applicant respectfully submits that Examiner is erroneously equating virtual address translation with page faults.

2. Nachos eliminates the page table and replaces it with a TLB. In Kaiser, a TLB miss is an address translation failure, and the subsequent page table miss is the page fault. But since Nachos replaced the page table with a TLB, then in Nachos, a TLB miss is both an address translation miss and a page fault.

To support Examiner's position, Examiner has cited (in the Advisory Action) a reference ("Nachos") that refers to address translation using a tag called "pagefaultexception". Examiner uses this command name to argue that the term "page fault" must include virtual address translation.

When a unit that is not a host processor needs data that has a virtual address, it must do two things, as described above. First, it must translate that address. This is done in Kaiser by making a comparison using the TLB. If the TLB is outdated, then Kaiser's memory controller goes to the host processor's page table to get the address translation. However, if the host processor's page table is outdated (a "page table miss"), then this means that not only is the TLB translation incorrect, but the data is not accessible to the host processor in main memory. This is a page fault.

The Nachos reference does not distinguish between these two actions for two reasons: first, because Nachos itself is the host, and second, because Nachos replaces the Nachos (host) page table with a TLB. Nachos refers to using a TLB in place of a page table to find an address translation, which gives the location in memory of the data. Nachos states at page 1, under the heading "The TLB":

> Up until now, Nachos has used a page table to translate from a virtual address to a physical address. This project switches from the page table to a TLB.... Nachos uses the TLB to convert from the virtual address that it receives into a physical address in order to actually access Nachos main memory.

[Emphasis added.]

In sating "Nachos main memory", this means that Nachos is the host. Nachos goes on to say:

> The problem with the TLB, just like the page table, is updating it. With a page table, it was simple: whenever there was a context switch, the machine -> page Table pointer was updated to point to the page table

> of the thread taking over the CPU. A TLB isn't quite as simple. There's only the one single TLB for the entire instance of Nachos, and Nachos itself has no clue what to do when it can't find something in the TLB. When this happens, it throws a PageFaultException.

[Emphasis added.]

Hence, where Nachos says, "when it can't find something in the TLB", it is referring to a page fault condition. However, it is important to note that Nachos has replaced the page table with the TLB. In short, where Kaiser uses a TLB for address translation and then (if address translation fails) it uses the host's page table to access main memory, Nachos has no host page table to which it can refer--Nachos itself is the host, and its TLB has replaced its page table.

It is noted that Nachos does not teach or suggest virtual address translation or page faulting without host processor involvement. This is because Nachos itself is the host. Applicant notes that Examiner does not characterize Nachos as teaching page faults without host processor involvement. Examiner only cites Nachos to argue that, because Nachos describes a TLB miss as a page fault, then the TLB miss in Kaiser must also be a page fault.

As argued above, Applicant respectfully submits that this is incorrect, since Nachos replaced its (i.e., the host) page table with a TLB (which Kaiser ONLY uses for address translation). Hence, the TLB miss of Nachos is actually the same as a host page table miss from Kaiser.

In the Advisory action, Examiner states in part:

> Applicant argues that the TLB miss described by Kaiser does not read on a "page fault". However, after researching the broadest reasonable definition of a "page fault", examiner has concluded that a TLB miss does meet this definition.

Hence, Examiner uses Nachos to argue that any TLB miss can be read as a page fault. In response, Applicant respectfully points to the argument presented above, to wit, that the TLB miss in Kaiser is an address translation miss, because Kaiser uses the TLB for address translation. Kaiser explicitly states that it uses the host's page table in the event of an address translation failure, and if the data is not in the host's page table, then a page fault occurs. But the TLB miss in Nachos is in fact a page fault, because Nachos itself is the host and replaced the page table with a TLB.

This distinction arises because the TLB in Kaiser is on a graphics processor, and is used by a memory controller. Kaiser's host processor still has a page table. But

in Nachos, Nachos itself is the host, and it normally would have a page table, but instead uses the mechanism of a TLB to replace the page table. Hence, Kaiser's TLB miss is merely an address translation miss, and the page fault doesn't occur until the host's page table is checked. But in Nachos, the TLB miss is the same as checking the host's page table--Nachos is the host, and it uses a TLB instead of a page table.

Hence, Examiner's argument that Kaiser's page fault includes a TLB miss is incorrect. As stated above, Kaiser distinguishes between the TLB miss (i.e., an address translation failure) from a page table miss (which is a page fault).

3. Nachos Reference Not Cited until Advisory Action

It is noted that the new Nachos reference was only presented in an advisory action and not in any Office Action to which Applicant could respond. It is respectfully submitted that reliance on this reference is therefore improper. 37 CFR 1.104(c) requires that the Examiner cite the best references available in rejecting claims. Since the Nachos reference was not cited in an Office action, nor presented to Applicant nor mentioned by Examiner until the Advisory Action, Applicant respectfully submits that Examiner may not rely on the Nachos reference in rejecting the claims.

B. Claims 3, 14, and 15

1. Kaiser does not appear to show a dedicated graphics memory, as claimed in claims 3, 14, and 15.

The arguments presented in favor of group A, above, are believed to apply also to claims 3, 14, and 15. These claims are also believed separately patentable, based on the following arguments.

The memory hierarchies are arranged differently in the Kaiser reference than in the present application. In the present application, as exemplified in claims 3, 14, and 15, there is a dedicated graphics memory, a system memory, and the disk. In the Kaiser reference, there is only the system memory and the disk. Kaiser does not appear to include dedicated graphics memory. Therefore, it is respectfully submitted that the Examiner has failed to make out a *prima facie* case of obviousness.

This distinction not only shows that the Examiner has not made a *prima facie* case of obviousness against these claims, it also shows important distinctions between the teachings and context of Kaiser compared with the present application.

As an example for purposes of discussion, claim 3 is reproduced below:

3. A computer system, comprising:

> at least one CPU, operatively connected to have read/write access to a main memory;
> first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit; and
> a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU.

In rejecting, for example, claim 3, Examiner states:

> As to claim 3, Kaiser discloses: at least one CPU, operatively connected to have read/write access to a main memory (fig. 2, the CPUs have access to the memory through a controller); first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit (col. 1, lines 13-45, virtual addressing on a hard disk is disclosed); a graphics accelerator unit, comprising dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU (see rejections to claims 1 and 2 and also the page buffer of figure 2 for a dedicated graphics memory).

The page buffer of figure 2, to which Examiner refers, is shown within the confines of memory controller 204 in figure 2. It is mentioned at column 5, line 58 as storing the address translations as taken from the page tables of system memory 24:

> ...memory controller 204 performs a table walk of page tables in memory system 24.... Memory controller 204 stores current copies of system architected facilities used to walkt he system page tables when doing address translation. Memory controller 204 fetches cache lines from the system page table on an as-needed basis to find an effective to real address translation. If no entry is found in the system page table 226, a page fault condition is sent to the graphics engine 206.

[Emphasis added.] This passage is reproduced to show that page table buffer 226, which Examiner equates in the rejection with the claimed "dedicated graphics memory," of (for example) claim 3, is not in fact a dedicated graphics memory. Instead, it is a part of the memory controller where address translations are stored,

after having been read from the main memory 24. A dedicated graphics memory, on the other hand, stores data associated with graphics processing, for example, texture data, or other data used by a graphics processor. A dedicated graphics memory is used primarily or only by a graphics processor. However, in Kaiser, the page table buffer 226 is used by the memory controller, which controls access by the host CPU to memory 24.

2. Kaiser does not appear to teach or suggest a second memory management unit as claimed in claims 3, 14, and 15.

Additionally, claims 3, 14, and 15 are believed separately patentable on separate grounds. The Kaiser reference fails to teach or suggest a second memory management unit, as claimed, for example, in claim 3. The rejection of claim 3 has been reproduced above. In the rejection, Examiner mentions the second memory management unit, but only refers back to the rejection of claim 1 and 2 for teaching in Kaiser of the second memory management unit. Neither claim 1 nor claim 2 claims a second memory management unit, and the rejections of claims 1 and 2 do not point to any teaching in Kaiser of a second memory management unit. Examiner also refers to the page buffer of figure 2 of Kaiser, but this cannot be a second memory management unit.

The only memory management unit in Kaiser is the memory controller 204, which controls memory accesses by CPU 12 to memory system 24. No other memory controller is found in the teaching of Kaiser, and Examiner relies on no other reference for such teaching. Therefore, it is respectfully submitted that Examiner has not made out a *prima facie* case of obviousness against at least claims 3, 14, and 15.

The second memory management unit is not a trivial element, and shows important distinctions between the Kaiser reference and the present claims. For example, in at least claims 3, 14, and 15, the inventions are claimed in the context of a separate graphics accelerator unit. This separate graphics accelerator unit has its own dedicated graphics memory, which in turn has its own memory management unit, as claimed. This support for an extra level of memory hierarchy (not shown in Kaiser) allows the host processor's virtual memory system to be leveraged, to increase the amount of texture storage beyond what the system memory can support or allow to be dedicated to this function efficiently. These advantages are not taught or suggested in Kaiser, because they are not present.

All grounds of rejection are therefore believed addressed.

Requested Relief

For the reasons advanced above, Appellant respectfully contends that claims 1-5, 7-10, and 12-22 are patentable. Therefore, reversal of this rejection is respectfully requested.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection of this paper, including extension of time fees, to Deposit Account 07-2320 and please credit any excess fees to such deposit account.

Date: July 19, 2007

Respectfully submitted,

Patrick C.R. Holmes
Registration No. 46,380
Attorney for Appellant

Groover & Holmes
Customer No. 21906
P. O. Box 802889
Dallas, Texas 75380
Phone: 972.980.5840
Fax: 972.980.5841

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	David R. Baldwin
Application No.:	09/591,225
Filing Date:	June 9, 2000
Examiner:	Richer, Aaron M.
Title:	Graphic Memory Management with Invisible Hardware-Managed Page Faulting
Art Unit:	2671
Docket No.:	TD-155

APPENDIX A – CLAIMS APPENDIX

1. (original): A computer system, comprising:

a graphics accelerator unit which manages page faulting of texture data invisibly to the host processor.

2. (previously presented): A computer system, comprising:

a graphics accelerator unit which manages page faulting of texture data, from main memory used by at least one host processor into a dedicated graphics memory, invisibly to the host processor, except when said graphics accelerator unit calls for data which has not recently been present in said main memory.

3. (original): A computer system, comprising:

at least one CPU, operatively connected to have read/write access to a main memory;

first memory management logic, which virtualizes said main memory with reference to at least one bulk storage unit; and

a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU.

4. (previously presented): A computer system comprising:

a host processor having respective physical memory associated therewith; and a graphics accelerator unit having respective local memory associated

therewith, and also having a graphics memory manager;

wherein, when said graphics accelerator unit attempts to access texture data which is in said physical memory associated with said host, said graphics memory manager fetches said texture data automatically.

5. (previously presented): The system of Claim 4, wherein, after fetching said texture data, said graphics memory manager restarts texture processing.

6. (canceled).

7. (previously presented): A computer system comprising:

a host processor having host physical memory associated therewith, and also having virtual memory management; and

a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and wherein, when said graphics accelerator unit attempts to access texture data which is in said host physical memory,

if said texture data is in said host physical memory, said graphics accelerator unit fetches said texture data there from automatically;

and if said texture data is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics accelerator unit fetches said texture data automatically from said host physical memory.

8. (previously presented): The system of Claim 2, wherein said graphics accelerator unit also includes a PCI/AGP interface, DMA controllers, SGRAM/SDRAM, a RAMDAC, and a video stream interface.

9. (previously presented): The system of Claim 2, wherein said dedicated graphics memory is SGRAM/SDRAM to which the unit has read-write access through its frame buffer and local buffer ports.

10. (previously presented): The system of Claim 2, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

11. (cancelled):

12. (previously presented): The system of Claim 3, wherein said first memory management logic is a bridge controller.

13. (previously presented): The system of Claim 3, wherein said bulk storage unit is a mass storage disk drive.

14. (previously presented): A computer system, comprising:

at least one CPU, operatively connected to have read/write access to a

main memory; first memory management logic, which virtualizes said main memory

with reference to at least one bulk storage unit; and a graphics accelerator unit, comprising rendering accelerator logic,

dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data, invisibly to said CPU;

wherein said graphics accelerator unit also includes a PCI/AGP interface, DMA controllers, SGRAM/SDRAM, a RAMDAC, and a video stream interface.

15.(previously presented): A computer system, comprising:

at least one CPU, operatively connected to have read/write access to a main memory; first memory management logic, which virtualizes said main memory

with reference to at least one bulk storage unit; and a graphics accelerator unit, comprising rendering accelerator logic, dedicated graphics memory, and a second memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data,

invisibly to said CPU; wherein said second memory management unit also manages texture storage in the main memory in addition to managing texture storage in normal texture memory.

16. (previously presented): The system of Claim 4, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

17. (previously presented): A computer system comprising:

a host processor having respective physical memory associated therewith; and a graphics accelerator unit having respective local memory associated

therewith, and also having a graphics memory manager; wherein, when said graphics accelerator unit attempts to access texture

data which is in said physical memory associated with said host,

said graphics memory manager fetches said texture data automatically; and wherein said graphics accelerator unit also includes a PCI/AGP

interface, DMA controllers, SGRAM/SDRAM, a RAMDAC, and

a video stream interface.

18. (previously presented): The system of Claim 4, wherein said graphics memory manager comprises:

means for determining where a page is located in host physical memory; means for updating page tables for said page; means for downloading said page; and means for restarting texture processing.

19. (previously presented): The system of Claim 7, wherein said host processor is operatively connected to receive inputs from input devices through an interface manager chip which provides an interface to various ports and registers.

20. (previously presented): A computer system comprising:

a host processor having host physical memory associated therewith, and also having virtual memory management; and

a graphics accelerator unit having respective physical memory associated therewith, and also having virtual memory management; and wherein, when said graphics accelerator unit attempts to access texture data which is in said host physical memory,

if said texture data is in said host physical memory, said graphics memory manager fetches said texture data therefrom automatically;

and if said texture is not in said host physical memory, said texture data is first loaded into said host physical memory, and thereafter said graphics memory manager fetches said texture data automatically from said host physical memory; and wherein said graphics accelerator unit also includes a PCI/AGP interface, DMA controllers, SGRAM/SDRAM, a RAMDAC, and a video stream interface.

21. (Previously presented) The system of claim 3, wherein said accelerator unit determines where the page is located in host physical memory to be manipulated is located; and wherein said accelerator unit determines which page out of the working set to use, marks this page the most recently used page, and updates the page tables for the new page and remove any reference to the page just bumped out of memory.

22. (Previously presented) A graphics accelerator unit, comprising:

a graphics accelerator unit, and a memory management unit which manages texture data for said accelerator logic and performs page faulting of said texture data; and

wherein said accelerator unit determines where the page is located in host physical memory to be manipulated is located; and wherein said accelerator unit determines which page out of the working set to use, marks this page the most recently used page, and updates the page tables for the new page and remove any reference to the page just bumped out of memory.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	David R. Baldwin
Application No.:	09/591,225
Filing Date:	June 9, 2000
Examiner:	Richer, Aaron M.
Title:	Graphic Memory Management with Invisible Hardware-Managed Page Faulting
Art Unit:	2671
Docket No.:	TD-155

APPENDIX B – EVIDENCE APPENDIX

None.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	David R. Baldwin
Application No.:	90/591,225
Filing Date:	June 9, 2000
Examiner:	Richer, Aaron M.
Title:	Graphic Memory Management with Invisible Hardware-Managed Page Faulting
Art Unit:	2671
Docket No.:	TD-155

APPENDIX C –RELATED PROCEEDINGS APPENDIX

None.

01-22-07

AF/$

PTO/SB/31 (04-05)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

NOTICE OF APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Docket Number (Optional): TD-155

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on January 19, 2007

Signature: Barbara Downs

Typed or printed name: Barbara Downs

Ex Mail EV 778057175

In re Application of: David R. Baldwin

Application Number	Filed
09/591,225	06/09/2000

For: Graphic Memory Management With Invisible

Art Unit	Examiner
2671	Aaron M. Richer

Applicant hereby appeals to the Board of Patent Appeals and Interferences from the last decision of the examiner.

The fee for this Notice of Appeal is (37 CFR 41.20(b)(1)) $ 500.00

- ☐ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee shown above is reduced by half, and the resulting fee is: $ _____
- ☑ A check in the amount of the fee is enclosed.
- ☐ Payment by credit card. Form PTO-2038 is attached.
- ☐ The Director has already been authorized to charge fees in this application to a Deposit Account. I have enclosed a duplicate copy of this sheet.
- ☑ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 07-2320. I have enclosed a duplicate copy of this sheet.
- ☐ A petition for an extension of time under 37 CFR 1.136(a) (PTO/SB/22) is enclosed.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

- ☐ applicant/inventor.
- ☐ assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)
- ☑ attorney or agent of record. Registration number 46,380
- ☐ attorney or agent acting under 37 CFR 1.34. Registration number if acting under 37 CFR 1.34. _____

Signature

Patrick C. R. Holmes
Typed or printed name

972-980-5840
Telephone number

January 19, 2007
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☐ *Total of 1 forms are submitted.

This collection of information is required by 37 CFR 41.31. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11, 1.14 and 41.6. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any

Applicant: David R. Baldwin
Title: Graphic Memory Management with Invisible Hardware-Managed Page Faulting
Docket No.: TD-155

Certificate under 37 CFR 1.10 of Mailing by "Express Mail"

EV701671240US
"Express Mail Label Number

July 19, 2007
Date of Deposit

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: MS Patent Application, Commissioner for Patents, PO Box 1450, Alexandria VA 22313-1450.

Carol Boultinghouse
Signature of person mailing correspondence

Carol Boultinghouse
Typed or printed name of person mailing correspondence

Enclosures:

1. Transmittal Form
2. Appeal Brief
3. Appendix A - Claims Appendix
4. Appendix B - Evidence Appendix
5. Appendix C – Related Proceeding Appendix
6. Previously Filed Notice of Appeal
7. Petition for Extension of Time
8. Cashier's Check No. 157724 in the amount of $1,590.00
9. Express Mail Certificate
10. Two (2) Return Post Cards